REMARKS

This amendment is responsive to the Office action made final mailed May 5, 2003 for the above-captioned application.

- Claims 13-16, 19 and 20 have been rejected under 35 USC 102(e).
- Claims 1-12, 17, 18 and 21 have been rejected under 35 USC 103(a).

Claims 1-6 and 13 have been amended. Claim 22 is added. No claims have been canceled. Claims 1-6 and 13-22 are pending.

The prior art rejections are respectfully traversed. Reconsideration is requested. Following are detailed comments on the cited art clarifying the distinctions between applicant's claimed inventions and the cited art.

OA ¶2-10: Claims 13-16, 19 and 20

The examiner has rejected claims 13-16, 19 and 20 under 35 USC 102(e) as being anticipated by U.S. Patent No. 6,173,389 (Pechanek et al.) Pechanek et al. disclose an indirect VLIW processor with a method for selecting subinstructions to be executed in parallel. The processor includes processing elements (PE). Each PE includes multiple execution units 131 (e.g., store, load, ALU, multiple and accumulate (MAU), data select (DSU)). Associated with each PE is a VLIW memory, (VIM).

Pechenak et al. strive to achieve a compressed VLIW memory and the ability to reuse instruction components, (Col. 2, lines 61-63). They compress VLIW memory by using the same VLIW instruction at different times, so that multiple copies are not stored. They reuse instruction components by taking a VLIW instruction and first masking off a portion so that not all of the PE's multiple units receive a subinstruction portion of the VLIW instruction. **Note however, that in no case does Pechanek take a given VLIW instruction in VIM and parse it, wherein one subinstruction portion is directed to multiple units for concurrent execution.**

Pechanek et al. include group control instructions which provide a degree of control in composing and executing VLIW instructions. Composing is achieved with LV instructions. Executing is achieved with XV instructions. With regard to the LV instruction, Pechanek et al. include a short instruction word (SIW) fetch controller which reads short instructions words from SIW memory, and **sequentially loads SIWs into VLIW memory to compose VLIW instructions**, (col. 2, line 64 to col. 3 line 4). The LV instruction is used to either load or disable individual instruction slots of a specified VIM address. For example, one LV instruction can disable slots, while a subsequent LV instruction loads SIWs. (Col. 6, lines 17-44). Note that the SIW controller is a short instruction word controller acting on short instructions words. Accordingly, it does not test, process or move VLIW instructions. It moves SIWs which are to become part of a VLIW instruction. It does not test control bits to determine whether to route a subinstruction portion of a VLIW instruction to multiple functional units.

With regard to the XV instructions, there is an XV1 instruction and an XV2 instruction. The XV1 instruction relates to compression across instruction slots, (col. 6, lines 12-13). The XV2 instruction relates to compression within an instruction slot (col. 8, lines 64-65). A slot refers to a channel for moving a subinstruction.

An XV1 instruction is used to modify, enable/disable sub-iVLIW instructions, and indirectly execute iVLIW instructions. (Col. 6, lines 14-16). Referring to Pechanek Fig. 5, the XV1 instruction includes VIM offsets, an XV1 opcode, and mask enable bits. The VIMOFFs bits are used to derive the VIM address 511. The XV1 opcode is decoded to generate the IR2MUX 1 control signal and the Xvc1 control signal, which set up the multiplexors 530-538 to select VIM output paths 541 to 549 (col. 7, lines 26-30). As a result, the subinstructions from the addressed VIM VLIW instruction are passed to the IR2 units 514. The mask enable bits 10-17 determine whether the instruction slot is to be active. More specifically, the mask enable slots determine whether the active/inactive status of the slot previously set by an LV instruction is to be overridden. Note that an XV1 instruction does not route a subinstruction of a VLIW instruction to multiple functional units which process the VLIW instruction. As there is no subinstruction sharing, there are no control bits to direct subinstruction sharing.

The XV2 instruction also is used to modify, enable/disable sub-iVLIW instructions and indirectly execute iVLIW instructions. The difference between the XV1 and XV2 instruction is evident from a comparison of Figs. 7 and 9. In Fig. 7, the XV1 instruction allows the same iVLIW instruction at VIM address 0 to be re-used later by masking off a subinstruction portion, as shown by comparing clock cycles 1, 3 and 7 of instruction sequence 720. In Fig. 9 the XV2 instruction uses separate subinstruction addresses to achieve an iVLIW instruction to be executed. (Col. 11, lines 12-24). In a sense, the resulting iVLIW instruction is composed of subinstructions from one or more VLIW instructions in VIM. Note that for a given VLIW instruction there is no sharing of a subinstruction among multiple functional units.

Of significance is that **Pechanek et al. re-use subsinstructions in different VLIW instructions to be executed at different times** (see the instructions execution cycles 720 and 920 of Figs. 7 and 9 and the related description). **This is different than the subinstruction sharing claimed by applicants in which a subinstruction from a given VLIW instruction is currently allocated to multiple functional units for concurrent execution**. It is apparent that Pechanek does not teach or suggest such subinstruction sharing. Notice that the subinstructions relating to the store unit cannot be shared with the load ALU, MAU or DSU units in the PE for concurrent execution. These execution units are functionally distinct making concurrent subinstruction sharing unfeasible.

With regard to claim 13, applicants recite that an instruction of VLIW architecture includes up to a second prescribed number of subinstructions, where that second number is the number of clusters times the common number of functional processing units in each cluster. Pechanek et al. disclose a system in which the VLIW instruction includes a prescribed number of subinstructions, where the prescribed number is equal to the common number (5) of execution units within a given PE. To be read on applicant's claim 13, there would need to be up to 4 (the number of PEs) times 5 (the number of execution units per PE), which is equal to 20 subinstructions in a VLIW instruction. Pechanek has 5 subinstructions per iVLIW for every instruction. Applicant's VLIW architecture allows for VLIW instructions to be relatively longer. Applicant shares subinstructions so that the VLIW instruction as stored in main memory and/or cache can be shortened.

Claim 13 distinguishes over the cited art based at least upon the following claim limitations:

a processor having a very large word instruction architecture ... said very large word instruction architecture allowing <u>an instruction to have up to a second prescribed number of subinstructions</u>, where the second prescribed number equals the first prescribed number times the common number, each instruction to be executed by the processor comprising from one subinstruction up to the second prescribed number of subinstructions, along with a set of control bits; and an instruction cache memory which stores a first instruction in a compressed format determined by a condition of the set of control bits, the compressed format including a shared subinstruction stored in a given field of the first instruction which is to be shared by a plurality of the functional processing units, said plurality of functional processing units being determined by said condition of the set of control bits.

Pechanek does not disclose control bits within a given VLIW instruction which determine how one subinstruction of a VLIW instruction already stored in cache is to be distributed to multiple functional processing units. **Specifically, the LV and XV instructions do not include subinstruction fields. The iVLIW instructions do not include control bits that direct a subinstruction to be distributed to multiple functi0onal units. An XV instruction in combination with an iVLIW instruction does not take a subinstruction of the iVLIW and distribute it to more than one functional processing unit.**

Claims 14-16 and 20 distinguish over the cited art based on the same reasons as given for claim 13, and further distinguish over Pechanek by reciting that a subinstruction of the given VLIW instruction is shared between functional processing units of different clusters.

Claim 20 further distinguishes over the cited art based on the following claim limitations:

- in which a first instruction in uncompressed format includes the second prescribed number of subinstructions;
- means for reducing the size of the first instruction
- means for loading the first instruction into the instruction cache in compressed format.

Pechanek includes 5 subinstructions in every iVLIW instruction and thus does not include the second prescribed number of instructions (which in Pechanek would be 20). Pechanek does not take a first instruction and reduce its size. Pechanek starts with short instruction words and compose VLIW instructions. The VLIW instructions are not reduced in size. Each one is the same length. Pechanek does not load a compressed VLIW instruction into instruction cache. The iVLIW loaded into instruction cache is not compressed. It may be a re-used iVLIW instruction but it is not a compressed VLIW instruction. (Pechanek compresses their program length and their instruction memory requirement, but do not compress their individual VLIW instructions as stored in the instruction cache).

OA ¶ 11-20: Claims 1-3 and 17

Claims 1-3 and 17 have been rejected under 35 USC 103(a) as being unpatentable over Pechanek et al., in view of the publication of Rosenberg.

Claim 1 recites a method for sharing subinstructions of a given VLIW instruction among functional processing units of multiple clusters on a VLIW processor. When the control bits of the given instruction identify a prescribed condition, a subinstruction is routed to multiple functional processing units as determine by the prescribed condition.

As previoulsy distinguished regarding claim 13, Pechanek does not use the control bits of a given instruction to route the subinstructions of such VLIW instruction. Pechanek uses a group control instruction to control processing of an iVLIW instruction. The group control instructions do not alter the routing of composed VLIW instructions. The FV2 instruction controls loading of subinstructions without sharing subinstructions among functional processing units to execute concurrently. The FV1 instruction does nonvaried loading of an iVLIW's subinstructions and controls whether each given subinstruction slot for the instruction is active or inactive.

The examiner cites Pachanek col. 6 lines 43-44 and col. 9, lines 24-33 and element 455 of Fig. 4C as disclosing control bits. Claim 1 recites that the set of control bits determine how a shared subinstruction is routed. The Pechanek passage at col. 6 lines 43-44 is for a d-bit. The d-bit is to enable or disable a subinstruction slot. This determines whether the subinstruction loaded in the slot will be executed. It does not alter routing of the subinstruction, nor allow for redundant routing of subinstructions of the current VLIW instruction. The Pechanek passage at col. 9, lines 24-33 describes the bit functions in the LV instruction. Such bits encompass an opcode, 2-bits relating to the slot disable function, 3 bits indicating the number of SIWs to follow, and 10-bits of an VIM address offset at which to store those SIWs. There is no description of any bits that would determine how one of those subsequent SIWs is to be distributed to multiple functional units of a given execution of a VLIW instruction.

The examiner cites Pechanek at col. 10, lines 19-22 and 44-54 as disclosing testing the control bits. Lines 19-22 relate to the decoder unit 812 (Fig. 8 therein) which generates signals IR2MUX1 and VXC1. As discussed above in summarizing the XV instructions these outputs are used to cause the IRS multiplexors to select the VIM output paths. Lines 44-54 relate to the operation of the XV2 instruction. The examiner asserts that this supports subinstruction sharing. It is respectfully submitted that a careful reading of the passage reveals that this is not subinstruction sharing. The section is better understood after reviewing Fig. 9 and the description of the instruction execution cycles 920 described at col. 11, lines 12-24. A subinstruction is not shared across different functional units, nor across different functional units for a current VLIW instruction. The XV2 instruction allows a subinstruction to be re-used for a later VLIW instruction **at the same functional unit**. Given the disclosure of being re-used at the same functional unit, there can be no suggestion of the subinstruction being concurrently executed at different functional units. Such an interpretation is based only on hindsight.

The examiner indicates that in parallel processing methods, instructions are executed concurrently in different processing units of different clusters. However, neither Pechanek nor Rosenberg teach that any of those subinstructions being concurrently executed are shared subinstructions that have been routed from one common subinstruction of a given source VLIW instruction.

Claims 2 and 3 depend from claim 1 and distinguish over the cited art for the same reasons as given for claim 1. Claims 2 and 3 further distinguish over the cited art based on the same reasons given in regard to claim 14 above.

Claim 17 depends from claim 14 and distinguishes over the cited art for the same reasons as given for claims 13 and 14. Claim 17 further distinguishes over the cited art for the same reasons as given for claim 1 above.

OA ¶22-26: Claims 4-5

Claims 4-5 have been rejected under 35 USC 103(a) as being unpatentable over Pechanek in view of U.S. Patent No. 6,044,450 (Tsushima), and further in view of U.S. Patent No. 5,819,058 (Miller et al.).

Pechanek does not disclose a VLIS instruction having a first prescribed number of subinstructions, where the first number is equal to the number of clusters times the common number of functional units in each cluster. Pechanek discloses VLIW instructions having a number of subinstructions equal in every VLIW instruction to the number of functional units (5), not the number of PE's (4) times the number of functional units (5) – which would be 20 subinstructions per VLIW instruction. Accordingly, Pechanek discloses a different architecture. Such difference is significant because applicants perform subinstruction sharing to decrease the length of a current VLIW instruction as stored in instruction cache so that the instruction length is compressed. Pechanek disclose a method of re-using subinstructions in subsequent instructions so that fewer (not shorter) VLIW instructions are stored in memory.

Claim 4 is in independent format and distinguishes over the cited art based upon steps performed during compilation of the computer program. Neither Pechanek nor Tsushima disclose:

- during compilation of the computer program, the steps of:
- identifying a pattern in which a subinstruction occurs more than once in a given instruction...;
- determining whether the pattern is among a set of prescribed patterns; and
- ... setting a set of control bits for the instruction to indicate that the pattern is present.

Claim 5 distinguishes over the cited art for the same reasons as given for claim 4, and further based on at least the following limitations:

- during compilation of the computer program, compressing the given instruction when the pattern ... by deleting one occurrence of the redundant subinstruction ...

The examiner cites Pechanek as disclosing the first prescribed number of subinstructions in a VLIW instruction. However, it is clear that the Pechanek iVLIW instructions as stored in memory and cache include a fixed number of subinstructions. The fixed number is equal to the number of processing units in a PE. This corresponds to applicant's common number of functional units and not to the required first prescribed number. With regard to the assertion that Pechanek teaches setting the control bits to indicate a pattern of subinstruction distribution, see the comments cited above regarding claims 1 and 13.

The examiner cites Pechanek at col 3 lines 54-56 as disclosing deleting one occurrence of a redundant subinstruction to achieve a compressed subinstruction. The examier relies on the XV2 instruction as disclosing such feature. A careful reading of the XV2 description however shows that there is no deletion of redundant subinstructions within a given VLIW. Pechanek's LV2 instruction loads subinstructions into VIM at individual subinstruction addresses. Accordingly the VIM does not yet store VLIW instructions. If the examiner would read the subinstructions at a VIM VLIW address as a VLIW for such case, then there still are 5 subinstructions stored at such VLIW address. Thus the VLIW length has not been shortened. This is because Pechanek relates to re-using subinstrucitons for the same subinstruction slot (i.e., the same functional unit) at different times for different VLIW instructions. It is noted that the 5 subinstructions stored in VIM for an XV2 need not be for the given VLIW instruction to be executed.

The examiner also cites Tsushima as disclosing subinstruction sharing. This finding is respectfully traversed. Tsushima uses a code to identify a group of subinstruciton opcodes as stored in memory. The group code is not a subinstruction. The group code points to a location in an instruction table where a corresponding group of subinstructions are stored. (Col. 8, lines 40-50). All opcodes corresponding to the code are present in the table. Further, there is no disclosure of a group code signifying that there is one corresponding opcode stored in the table to be spread out for distribution to multiple functional processing units for processing of the current VLIW instruction.

OA ¶27-28: Claim 6

Claim 6 is in independent format and has been rejected under 35 USC 103(a) as being unpatentable over Pechanek in view of Tsushima and further in view of Rosenberg., and further in view of Miller. Claim 6 distinguishes over the cited art for the same reasons as given above for claim 1.

OA ¶29-33: Claims 18 and 21

Claims 18 and 21 depend from claim 14 and distinguish over the cited art based at east on the same reasons as given for claims 13 and 14.

New Claim 22

New claim 22 is based on the subject matter of claim 1 and includes select elements of claims 5 and 6. No additional issues are presented. No new subject matter is introduced. Claim 22 clarifies that the subinstruction routing and subinstruction sharing pertains to a current VLIW instruction to be processed by the group of clusters and their functional processing units. Claim 22 also clarifies that the instruction as stored in the instruction cache is a compressed-length VLIW instruction. Pechanek and the remaining cited art do not disclose a compressed length VLIW instruction in instruction cache, nor the sharing of a subinstruction within such current instruction among different functional processing units.

Conclusion

In view of the above remarks regarding the cited art, it is respectfully submitted that the claims contain key limitations that are not present in the cited art and not obvious from the cited art. These particular limitations, are not disclosed in or suggested by cited references. These limitations are significant advances over the prior art and resulted in a novel method and apparatus for sharing VLIW subinstructions.

In view of the above amendments and remarks, it is respectfully submitted that the claims are now in condition for allowance. The Examiner's action to that end is respectfully requested. Reconsideration of the claims and withdrawal of the rejections is respectfully requested.

If, in the opinion of the Examiner, a telephone conference would expedite the prosecution of the application, the Examiner is invited to call the undersigned attorney at the telephone number given below.

Respectfully submitted,

Dated: 8/5/03

By: [illegible]
Steven P. Koda
Reg. No. 32,252
Tel.: 845-689-2044

Koda Law Office
75A Lake Road, No. 365
Congers, N.Y. 10920